2021 FALL

# ECE 538 - Advanced Computer Architecture
## Homework Assignment 2 (Individual)

**100 points**

**Assigned Date:** 10/11/2021 **Due Date:** 10/21/2021

1. **(20 points)** Answer the following qualitative questions:

   (a) (5 points) What are the types of cache misses discussed in lecture so far? Briefly describe each.

1. Compulsory misses are the initial miss due to a cold cache. They exist as blocks are first brought into the cache.
2. Capacity misses are due to the cache being too small and exist when a cache cannot contain the entire program.
3. Conflict misses are due to the lack of associativity and occurs because the rigrid block placement strategy.
4. The fourth is coherency misses. We have talked about them but they exist when caches must be kept coherent because memory is being shared among processors.

   (b) (5 points) How can pipelining the L1 cache improve processor performance? What negative effects can it have?

pipelining L1 cache allow for a higher clock rate and a larger L1. the higher clock rate can improve instruction throughput and the larger L1 can reduce miss rate. The negative effects are the increased control and data hazards, which interrupt the flow of instructions through the pipeline. Besides, it will increase cache complexity.

   (c) (5 points) How does a direct-mapped cache compare to a set-associative cache in terms of access time, logic complexity, energy, etc.?

A directed-mapped cache is the simplest cache in terms of logic complexity and generally has a lower access time than set-associative. Although we have seen solusions where 2-way set-associtive was slightly faster than the directed-mapped. In terms of energy, each access for direct-mapped is almost always lower. However, direct-mapped suffers from more conflict miss.

   (d) (5 points) What is Average Memory Access Time (AMAT)? And how it can be calculated?

AMAT is the Average memory Access Time used to measure the cache organization.

AMAT = Hit Time + Miss Rate × Miss penalty.

2. **(25 points)** Assume you have an *inorder* RISC processor that has a CPI of 1.3 with an ideal memory hierarchy (i.e., all memory accesses hit in the L1 caches). For a given benchmark, 38% of instructions executed are loads and stores. The miss penalty for reads and writes to the L1 data cache is 20 clock cycles and the miss rate is 3%.

(a) (5 points) Calculate the CPI of your RISC processor taking memory stalls due to data accesses into account.

Because the processor is inorder, we can simply assum the processor is stalled during a cache miss. Thus, the CPI is directly impacted as shown below:

$$CPI = 1.3\ cc/instr + 38\% \times 3\% \times 20\ cc = 1.528\ cc/instr$$

(38% = access/instr, 3% = misses/access.)

(b) (5 points) The miss penalty for the L1 instruction cache is also 20 clock cycles and the miss rate is 1%. Recalculate the CPI of your processor taking into account stalls from both the instruction and data caches. How much faster is the machine with the ideal memory system?

$$CPI = 1.528\ cc/instr + 1\% \times 20cc = 1.728\ cc/instr.$$

(c) (5 points) Assuming your processor runs at 1 GHz, what is the average memory access time of the L1 instruction cache? What is the average memory access time of the L1 data cache?

We must assum that the L1 caches are accessed in 1 cc

$$AMAT_{instr} = 1cc + 1\% \times 20cc = 1.2\ cc$$

$$AMAT_{L1} = 1cc + 3\% \times 20\ cc = 1.6\ cc$$

(d) (5 points) Why might the hit rate be higher for the instruction cache compared to the data cache?

Instruction access has a lot more spatial and temporal locality than data access. this is because of the way the instructions are fetched sequentially and because function calls and loops.

(e) (5 points) Calculate the *misses per instruction* for both instruction and data caches. How does this metric differ from *miss rate*?

The misses per instruction for the instruction cache is the same as the miss rate because the instruction cache is accessed for every instruction. So, 1% miss/instruction.

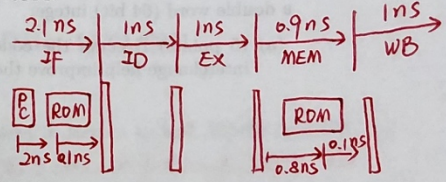
For the data cache, the misses per instruction is the misse rate multiplicated by the number of accesses per instruction. So, 3% miss/access × 38% access/instr = 0.014 mss/instr

Notice that the misses per instruction for the instruction and data cache are much closer.

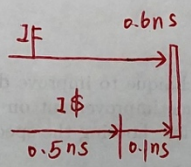
3. **(30 points)** You are designing a System On a Chip (SoC) with your 5-stage pipelined RISC processor. The minimum system requirements are 512KB of program ROM and 16KB of RAM. For a given technology node, the critical path of your processor (ignoring memory access) is 1 $ns$. The access time for a 512KB ROM in the same technology is 2.0 $ns$, while the access time for a 16KB RAM is 0.8 $ns$.

(a) (5 points) Assuming the Fetch and Memory stages of your processor only access memory and the flip-flop overhead is 0.1 $ns$, what is the fastest achievable clock rate of your processor when directly fetching instructions from the program ROM?

the critical path of the processor without memory is 1ns. this means the maximum delay through the ID. Ex. or WB stage is 1ns. If we fetch right from Rom, the IF stage will take 2.1 ns and the new critical path. thus. the clock rate is: $\frac{1}{2.1\,ns} = 476.2\,MHz$




(b) (5 points) Assuming accesses to program ROM are cached in a 4KB instruction cache with an access time of 0.5 $ns$, what is the fastest achievable clock rate of your processor?




If we cache the ROM access, the IF stage will only take 0.6ns, so it is off the critical path. Thus, the processors can run at the full of 1.6GHz. $\left(\frac{1}{0.6ns}\right)$

(c) (10 points) If the miss rate of the 4KB instruction cache with a 16 byte block size is 6.3% for a given workload, what is the average number of memory stall cycles per instruction your processor will see with a 4KB instruction cache? Assume the instruction cache can start a read from program ROM on the cycle after the miss is detected, and critical word first is not implemented.

*Hint:* Start by calculating the miss penalty in clock cycles and remember a cache miss fills an entire line.

the miss penalty will be calculated as follows:

$$1\,cc + \left(\frac{16\ byte}{4\ byte}\right) \times \left[\frac{2.1\,ns}{1\,ns/cc}\right] = 1cc + 4\times 3cc = 13\,cc$$

- 1 cc: the first cycle when the miss is detected
- $\frac{16\ byte}{4\ byte}$: # of ROM reads to fill cache line
- $\frac{2.1\,ns}{1\,ns/cc}$: determine number of cycles to read the ROM.

(d) (10 points) Without the instruction cache, the processor is able to complete 4 million instructions from the same workload in 4.8 million clock cycles. What will be the CPI of your processor with the 4KB instruction cache, assuming the same workload?

without the instruction cache,

the CPI = 1.2 cc/instr + 6.3% × 13

= 2.02 cc/instr

thus, stalls per instruction:

= 6.3% × 13 cc

= 0.819 cc/instru

4. **(25 points)** Consider the following simple C code for matrix transpose:

```
for(i = 0; i < 1024; i++)
    for(j = 0; j < 1024; J++){
        Y[j][i] = X[i][j]
    }
}
```

the goal of blocking is to create blocks that are the size of an L1/L2 cache.

Assume that both X and Y are stored in the row major order and each element in the matrices is a double word (64 bit) integer.

(a) (5 points) Execute the code on your laptop and report the execution time. Would loop interchange help improve the performance of the code? Give explanation.

(b) (10 points) Rewrite the code by applying blocking (the software technique to improve data reuse). Use a blocking factor that leads to a noticeable performance improvement on the same computer as the one used in Question A. Explain your reason for choosing the specific blocking factor.

(c) (10 points) Execute the new code and report the execution time. Make a comparison with the results reported in 5-(a).